НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 “ Комп’ютерні науки”

на тему _Організація реєстру інформаційних ресурсів з використанням технології Blockchain

Виконав: студент ___4___курсу, групи_ТМ-52_

_____Осовецький Віталій Дмитрович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ___старший викладач Дацюк Оксана Антонівна_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “ Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ

на дипломну роботу студенту

Осовецький Віталій Дмитрович
(прізвище, ім’я, по батькові)

1. Тема роботи Організація реєстру інформаційних ресурсів з використанням технології Blockchain

керівник роботи старший викладач Дацюк Оксана Антонівна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”22”05 2019р. № 1325С

2. Строк подання студентом роботи 10.06.2019

3. Вихідні дані до роботи середовище розробки програмного забезпечення Visual Studio 2017, мова програмування C# та фреймворк ASP .NET Core

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Проналізувати інсуючі рішення для безпечного збереження власної інформації. Створити програмний продукт, який дозволяє безпечно та доступно маніпюлювати власними даними.

5. Перелік ілюстративного матеріалу
«Постановка задачі», «Існуючі рішення», «Використані технології», «Архітектура системи», Головні елементи інтерфейсу», «Програмні засоби реалізації», «Приклади інтерфейсу користувача», «Висновки».

7. Дата видачі завдання ”14” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	09.10.2018	
2.	Вивчення та аналіз задачі	14.10.2019-23.12.2018	
3.	Розробка архітектури та загальної структури системи	02.02.2019-03.03.2019	
4.	Розробка структур окремих підсистем	04.03.2019-14.04.2019	
5.	Програмна реалізація системи	15.04.2019-19.05.2019	
6.	Оформлення пояснювальної записки	20.05.2019-05.06.2019	
7.	Захист програмного продукту	14.05.2019	
8.	Передзахист	28.05.2019	
9.	Захист	17.06.2019-21.06.2019	

Студент _____ Осовецький В.Д.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Дацюк О.А.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи було створення програмного продукту для безпечного та доступного збереження власної інформації. Система надає можливість користувачу завантажувати власну інформацію в мережу, отримувати інформацію з мережі на власні носії та переглядати завантажені дані.

Розроблене рішення можливо буде використовувати в усіх сферах інформаційних технологій, в яких необхідне безпечне збереження даних.

Особливості архітектури програмного продукту передбачають можливість його використання як у якості настільного додатку, так і в контексті клієнт-серверної архітектури.

Записка містить 68 сторінок, 19 рисунків, 3 таблиці та 18 посилань.

ANNOTATION

The purpose of the work was to create a software product for the safe and affordable storage of their own information. The system should allow the user to download their own information to the network, receive information from the network on their own media and view the downloaded data.

The developed solution will probably be used in all areas of information technology that require secure data storage.

Features of the software architecture include the ability to use it as a desktop application, and in the context of client-server architecture.

The note contains 68 pages, 19 images, 3 tables and 18 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

TCP / IP – набір протоколів мережі Internet;

API – прикладний програмний інтерфейс;

ОС – операційна система;

CRM – управління відносинами з клієнтами;

CLI – інтерфейс командного рядка;

LINQ – запити інтегровані в мову;

IL – проміжна мова;

CTS – загальна система типів.

ВСТУП

В умовах швидкого розвитку наукоємних галузей людської діяльності, роль комп’ютерних технологій нестримно зростає. За останні роки значно збільшився потік інформації, як наслідок, з’явилася гостра потреба в нових способах її подання, формалізації, зберігання та систематизації, пошуку.

Інформаційне суспільство особливе значення надає інформації і інформаційним ресурсам. Інформація стала масовим товаром, який сьогодні доступний кожному. Інформаційні ресурси активно перетворюються в електронну форму.

Комп'ютерні системи і телекомунікації визначають надійність і потужність систем оборони і безпеки країни. Комп'ютери забезпечують збереження інформації, її обробку і надання її споживачам, реалізуючи в такий спосіб інформаційні технології.

Однак саме найвищий ступінь автоматизації, до якого прагне сучасне суспільство, ставить його в залежність від ступеня безпеки використовуваних ним інформаційних технологій, з якими пов'язані благополуччя і навіть життя безлічі людей. Стосовно інформаційних технологій це означає, що широке впровадження популярних дешевих комп'ютерних систем масового попиту і застосування робить їх надзвичайно вразливими щодо деструктивних впливів. Як показує аналіз останніх років, постійно винаходяться нові й нові види та форми обробки інформації і паралельно винаходяться все нові й нові види і форми її захисту, однак цілком її ніяк не вдається захистити.

Сучасні телекомунікаційні технології об'єднали локальні мережі в глобальні. Це привело до появи такого унікального явища, як Internet. І саме розвиток Internet викликав сплеск інтересу до проблеми безпеки і змусив, принаймні частково, переглянути її основні положення. Справа в тому, що, крім усіх плюсів користування, Internet забезпечує широкі можливості для здійснення порушень безпеки систем обробки інформації усього світу. Якщо комп'ютер підключений до Internet, то для

зловмисника не має ніякого значення, де він знаходиться – у сусідній кімнаті чи на іншому кінці світу.

Основною проблемою дипломної роботи є створення програмного продукту для безпечного збереження та доступ даних, що було б доступним та легким у використанні для всіх користувачі, що підключені до мережі Internet та не потребувало б специфічних знань криптографічних методів збереження інформації.

Розв’язком проблеми доступності є використання веб-технологій при розробці продукту та проблеми захисту інформації – підключення програмного продукту до мережі Blockchain.

Blockchain – це технологія розподіленого збереження даних в одноранговій мережі в вигляді безперервної послідовності блоків, зв’язаних між собою за допомогою алгоритму хешування.

Для розробки програмного забезпечення було використано мову програмування C#, фреймворки ASP.NET Core Web API, Bootstrap.

1. ЗАДАЧА РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ БЕЗПЕЧНОГО ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ

Метою роботи є створення програмного продукту для безпечного та доступного для користувача збереження власної інформації.

Для досягнення поставленої мети необхідно:

– проаналізувати вже існуючі програмні рішення;
– проаналізувати літературні джерела на відповідну тематику;
– обрати засоби та інструменти розробки програмного продукту;
– обрати методи розробки;
– розробити архітектуру програмного забезпечення;
– здійснити програмну реалізацію продукту для збереження інформації з використанням обраних технологій програмування.

Розроблений програмний продукт має включати наступний функціонал:

– реєстрація користувача в мережі;
– авторизація користувача в мережі;
– перегляд завантаженої інформації;
– завантаження інформації в мережу;
– отримання завантаженої інформації.

2. АНАЛІЗ ПРОБЛЕМИ БЕЗПЕЧНОГО ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ

У даній роботі було проведено аналіз проблеми безпечного та доступного для користувача збереження інформації та знайдено технологію, яка позволить реалізувати програмний продукт, який вирішить дану проблему.

2.1. Проблема безпеки власної інформації

Internet і інформаційна безпека несумісні по самій природі Internet. Вона народилася як чисто корпоративна мережа, однак, в даний час за допомогою єдиного стека протоколів TCP / IP і єдиного адресного простору поєднує не тільки корпоративні і повідомлюючі мережі (освітні, державні, комерційні, військові і т.д.), що є, за визначенням, мережами з обмеженим доступом, але і рядових користувачів, які мають можливість отримати прямий доступ в Internet зі своїх домашніх комп'ютерів за допомогою модемів і телефонної мережі загального користування.

Як відомо, чим простіше доступ у мережу, тим гірше її інформаційна безпека, тому з повною підставою можна сказати, що споконвічна простота доступу в Internet – гірше злодійства, тому що користувач може навіть і не дізнатися, що у нього були скопійовані – файли і програми, не кажучи вже про можливість їхнього псування і коректування [1].

Дилема безпеки така: доводиться робити вибір між захищеністю ваших даних і їх доступністю для вас, а значить, і можливістю корисного використання.

Це справедливо і щодо інформації. Наприклад, база даних, що містить конфіденційні відомості, лише тоді повністю захищена від зазіхань, коли вона знаходиться на дисках, знятих з комп'ютера і прибраних в захищене місце. Як тільки ви встановили ці диски в комп'ютер і почали використовувати, з'являється відразу кілька каналів, по яких зловмисник, в принципі, має можливість отримати до ваших

даних доступ без вашого відома. Іншими словами, ваша інформація або недоступна для всіх, включаючи і вас, або не захищена на сто відсотків [2].

Загрози безпеці даних та їх псування можна розділити на категорії, що їх спричиняють (Рисунок 2.1).

Рисунок 2.1 – Загрози безпеці даних

Серед усіх загроз найбільший відсоток за масштабом і ступенем пошкодження даних припадає на людський чинник.

Комп’ютерні віруси створюються людьми і розповсюджуються, як хвороба. Програма, до якої приєднався вірус, стає «зараженою».

Комп'ютерний вірус – це шкідлива програма, яка здатна до самокопіювання та може вбудовуватись у код інших програм, системні області пам’яті, завантажувальні сектори, а також поширювати свої копії різноманітними каналами зв’язку [3].

На сьогодні відомі мільйони різноманітних комп’ютерних вірусів. Усі їх умовно можна класифікувати за різними ознаками (Таблиця 2.1).

Таблиця 2.1 – Класифікації вірусів

Ознака	Типи вірусів	Принцип дії
Середовище існування	Файлові	Заражають виконувані файли й допоміжні програми
	Завантажувальні	Заражають завантажувальний сектор диска
	Макровіруси	Заражають файли Word, Excel тощо, які підтримують роботу макросів (вбудованих програм)
	Мережеві	Розповсюджуються мережею
Зовнішній вигляд	Звичайні	Програмний код вірусу видно на диску
	Невидимі	Програмний код вірусу не видно на диску
	Поліморфні	Програмний код вірусу видозмінюється, тому його складно виявити
Результати діяльності	Безпечні	Не виконують шкідливих дій, крім свого поширення і дратівливих ефектів
	Небезпечні	Виконують шкідливі дії, що призводять до пошкодження даних і програм

Програмного засобу, який би стовідсотково гарантував виявлення та знищення всіх вірусів, не існує. Боротися допомагають антивіруси — програми, призначені для виявлення та знешкодження відомих їм вірусів.

Існує багато антивірусних програм. Такі програми розрізняють за їхнім призначенням.

Детектори – виявляють файли, заражені вірусами.

Лікарі (фаги) – «лікують» заражені програми або документи, видаляючи в них віруси.

Ревізори – запам’ятовують стан програм і дисків, порівнюють їхній поточний стан із попереднім і повідомляють про виявлені невідповідності.

Фільтри – перехоплюють звернення до системи, які використовуються вірусами для розмноження і зараження файлів.

Блокувальники – перехоплюють вірусоподібну поведінку (відкриття файла програми для запису, запис у завантажувальний сектор диска тощо) і повідомляють про це користувача.

Сканери – одноразово сканують (перевіряють) усі файли на жорсткому диску та в оперативній пам’яті.

Імунізатори – імітують зараженість комп’ютера вірусом, щоб уник-іути реального зараження [4].

ОС Windows 8 і ОС Windows 10 мають вбудований антивірус Windows Defender (Захисник Windows), проте до складу більшості операційних систем антивірусні програми не входять.

До найпопулярніших комерційних антивірусних програм належать ESET NOD32, BitDefender тощо. Майже не поступаються за ефективністю безкоштовні програми 360 Total Security, Panda Free Antivirus, Avast Free Antivirus, Avira Free Antivirus, український антивірус Zillya!. Вони захищають не лише від вірусів, а й від троянських, шпигунських програм тощо.

2.2. Існуючі рішення

На даний момент існує багато хмарних сервісів для зберігання власної інформації, та кожна з них не дає гарантію на цілісність ваших даних, також можливе збереження на власних носіях, що не є досить зручним варіантом.

2.2.1. Amazon Web Services

Amazon Web Services (AWS) є дочірньою компанією Amazon.com, що надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної підписки. Існує і безкоштовна підписка, яка доступна протягом перших 12 місяців. Технологія дозволяє абонентам мати у своєму розпорядженні повноцінний віртуальний кластер комп'ютерів, який завжди доступний через Інтернет. Віртуальні комп'ютери AWS мають більшість атрибутів реального комп'ютера, включаючи апаратні пристрої (процесор, відеокарту, локальну та оперативну пам'ять, жорсткий диск або SSD-накопичувач); операційну систему на вибір; мережу; і попередньо встановлені прикладні програми, такі як веб-сервер, база даних, CRM і т. д. Кожна система AWS також віртуалізує консольний ввід/вивід (клавіатура, дисплей і миша), що дозволяє користувачам AWS підключитися до своєї системи AWS за допомогою браузера. Браузер виступає як вікно у віртуальний комп'ютер, дозволяючи користувачу входити в систему, налаштовувати та використовувати свої віртуальні системи так само, як справжній, фізичний комп'ютер. Це дозволяє їм налаштувати систему так, щоб надавати інтернет-орієнтовані сервіси та послуги своїм клієнтам [5].

Технологія AWS базується на серверних кластерах (фермах), розташованих по всьому світі. Плата за користування базується на комбінації використання апаратних засобів/ОС/програмного забезпечення/мережевих функцій, вибраних користувачем, а також вимог до доступності[en], надлишковості (redundancy), безпеки та додаткових параметрів. Виходячи з того, що користувач потребує і оплачує, він може зарезервувати один віртуальний комп'ютер (VM), кластер віртуальних комп'ютерів

(VM Cluster), фізичний (реальний) комп'ютер (Server), призначений для його виняткового використання, або навіть кластер фізичних комп'ютерів (Server Cluster). Компанія Amazon зобов'язується керувати та оновлювати програмне та апаратне забезпечення для дотримання необхідних стандартів безпеки. AWS працює в багатьох географічних регіонах, у тому числі в Канаді, Німеччині, Ірландії, Сінгапурі, Токіо, Сіднеї, Пекіні, Лондоні і т. д.

У 2016 році AWS надавав більш ніж 70 сервісів, що охоплюють широкий спектр, включаючи обчислення та зберігання даних, їхню передачу по мережі, аналітику, мобільні застосунки, інструменти для розробників і т. д. Найпопулярніші з них є Amazon Elastic Compute Cloud (EC2) і Amazon Simple Storage Service (S3). Більшість служб не надаються безпосередньо кінцевим користувачам, але замість цього пропонуються функціональні можливості через API, які розробники можуть використовувати в своїх програмах. Пропозиції Amazon Web Services доступні через HTTP, використовуючи архітектурний стиль REST та протокол SOAP.

2.2.2. Google Drive

Диск Google (Google Drive) – сховище даних, яке належить компанії Google Inc., що дозволяє користувачам зберігати свої дані на серверах у хмарі і ділитися ними з іншими користувачами в Інтернеті. Google Drive включає Google Docs, Sheets, and Slides, офісний пакет, який дозволяє спільно редагувати документи, електронні таблиці, презентації, малюнки, форми, і багато іншого.

Google Drive включає в себе систему обміну файлами, де автор файлу або папки за замовчуванням є його власником. Власник має можливість регулювати видимість файлу або папки для інших користувачів. Право власності підлягає передачі. Файли і папки можуть спільно використовуватись приватно конкретними користувачами, що мають обліковий запис Google, використовуючи свої @gmail.com адреси електронної пошти. Для спільного використання файлів з користувачами, які не мають облікового запису Google потрібно зробити файли «доступними за посиланням». Це створює секретний URL для файлу, який може бути відкритий через електронну пошту, блоги

і т. д. Файли та папки також можна зробити загальнодоступними в Internet, що означає, що вони можуть бути проіндексовані пошуковими системами і, таким чином, можуть бути знайдені і доступні будь-кому. Власник може також встановити рівень доступу. Три запропоновані рівня доступу, це «може редагувати», «може коментувати» і «може переглядати». Користувачі, що мають доступ для редагування можуть запрошувати інших редагувати.

Існує ряд зовнішніх веб-додатків («apps»), які працюють з Google Drive. Ці програми доступні у веб-магазині Chrome і сумісні з усіма підтримуваними браузерами. Щоб використовувати додаток, користувач повинен увійти в в Chrome Web Store і встановити додаток. Деякі з цих додатків розроблені Google, наприклад, Google Docs, Sheets and Slides. Додатки Drive, які працюють з інтернет-файлами, використовуються для перегляду, редагування і створення файлів в різних форматах, редагування зображення та відео, факсу і підписування документів, управління проектами, створення блок-схем, додатків і т. д. Drive також може бути програмою за замовчуванням для обробки файлів у форматах, які ним підтримуються. Деякі з цих додатків також працюють в автономному режимі, але лише на Google Chrome і Chrome OS. Усі сторонні додатки можна встановити безкоштовно. Тим не менш, деякі з них стягують додаткову плату за продовження використання або доступ до додаткових функцій. Більшість додатків Drive мають дозвіл на доступ до файлів користувачів за межами Google Drive. Збереження даних з сторонніх додатків на Google Drive вимагає авторизації в перший раз. Google Drive SDK працює спільно з Google Drive UI і Chrome Web Store, щоб створити екосистему додатків, які можуть бути встановлені в Google Drive.

Google Drive дозволяє переглядати в Інтернеті файли наступних форматів:

- вбудовані формати (Docs, Sheets, Slides, Forms, Drawings);
- файли зображень (.JPEG, .PNG, .GIF, .TIFF, .BMP);
- відео файли (WebM, .MPEG4, .3GPP, .MOV, .AVI, .MPEGPS, .WMV, .FLV, .OGG);
- аудіо формати (MP3, MPEG, WAV, .ogg);
- текстові файли (.TXT);

- розмітка / Код (.CSS, .HTML, .PHP, .C, .CPP, .H, .HPP, .JS);
- Microsoft Word (.DOC and .DOCX);
- Microsoft Excel (.XLS and .XLSX);
- Microsoft PowerPoint (.PPT and .PPTX);
- Adobe Portable Document Format (.PDF);
- Apple Сторінки (.PAGES);
- Adobe Illustrator (.AI);
- Adobe Photoshop (.PSD);
- Autodesk AutoCAD (.DXF);
- Scalable Vector Graphics (.SVG);
- PostScript (.EPS, .PS);
- шрифти (.TTF);
- специфікація XML Paper (.XPS);
- типи файлів архіву(.ZIP, .RAR, tar, gzip);
- .MTS Файли;
- формати необроблених зображень;

Файли в інших форматах можуть бути також оброблені за допомогою сторонніх додатків, які працюють з Google Drive і доступні у Chrome Web Store. Додатки Google Drive для Android і iOS можуть використовувати інші програми, встановлені на пристрої, щоб відкрити непідтримувані типи файлів [6].

Завантажені, але не перетворені у формат Документів Google, файли можуть бути до 5 ТБ в розмірі. Створені або завантажені файли розміром більше 5 ТБ не можна переглядати в Google Drive. Вбудовані зображення не повинні перевищувати 2 Мб кожне.

2.2.3. Dropbox

Dropbox – файловий хостинг компанії Dropbox Inc., що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.

Dropbox дозволяє користувачеві розміщувати файли на віддалених серверах за допомогою клієнта або з використанням веб-інтерфейсу через браузер. При установці клієнтського програмного забезпечення Dropbox на комп'ютері створює синхронізуючу папку. Хоча головний акцент технології робиться на синхронізацію і обмін інформацією, Dropbox веде історію завантажень, щоб після видалення файлів з сервера була можливість відновити дані. Також ведеться історія зміни файлів, яка доступна на період останніх 30 днів, крім цього доступна функція безстрокової історії зміни файлів «Pack-Rat».

Історія зміни файлів ведеться за принципом diff-кодування, щоб заощадити місце, займане файлами. В історії зміни записується тільки відмінність однієї версії файлу від іншої. Файли, завантажені через клієнт, не мають обмеження на розмір, але файли, завантажені через веб-інтерфейс, обмежені 20 ГБ. Є також можливість викладати файли для загального доступу через папку «Public», що дозволяє використовувати сервіс як файлообмінника. У версіях 0.8.x також з'явилася можливість надання в загальний доступ будь-якої папки в «My Dropbox» для подальшого доступу через так званий «shareable link», тобто через веб-інтерфейс. Для спільної роботи над проектами сервіс має можливість створення «Shared» папок для загального доступу осіб, які мають різні облікові записи на сервісі. Доступна автоматична синхронізація файлів і папок і зберігання версій з можливістю відкату. Для користувачів Dropbox Professional і Dropbox Business доступна функціональність Smart Sync дозволяє економити місце на жорсткому диску відображаючи тільки назви і інформацію про файли не завантажуючи на їх вміст.

На відміну від ряду аналогів, Dropbox не використовує шифрування даних на стороні клієнта, що, зокрема, зробило можливим інцидент 19 червня 2011 року, коли через помилку в оновленому програмному забезпеченні сервера протягом чотирьох годин був можливий вхід в будь-який аккаунт з використанням будь-якого пароля.

Сервіс пропонує безкоштовно 2 ГБ для зберігання даних, які можна збільшити безкоштовно до 16 ГБ, запрошуючи нових користувачів або ж отримати кілька гігабайт після виконання завдань (установка програми Dropbox на мобільний телефон і т. д.), також можна купити 1 ТБ.

Є офіційне SDK для створення власних додатків під Dropbox з використанням популярних мов і платформ Swift, Objective-C, Python, JavaScript, Java, HTTP, .NET.

2.2.4. OneDrive

OneDrive – це файловий хостинг, що надається компанією Майкрософт як частина набору онлайн-послуг. Він дозволяє користувачам зберігати файли, а також інші особисті дані, такі як налаштування Windows або ключі відновлення BitLocker у хмарі. Файли можна синхронізувати з ПК та отримувати доступ до них з веб-браузера або мобільного пристрою, а також ділитися публічно або з певними людьми.

Характеристики:

– сервіс OneDrive дозволяє зберігати до 5 ГБ інформації безкоштовно;
– для зображень передбачений попередній перегляд у вигляді ескізів, а також можливість їх перегляду у вигляді слайдів;
– для користувачів Windows 8 доступно 25 ГБ;
– для всіх папок і файлів можна визначити рівень доступу – від виключно персонального до публічного;
– є недокументований доступ за протоколом WebDAV;
– випускаються клієнтські додатки для Android, iOS, Windows Phone, Windows, Xbox (в тому числі Windows 8), OS X, MeeGo 1.2 Harmattan, Symbian Belle;
– для бізнесу може бути отримана автономна версія;

Існує підтримка Office Online в OneDrive. Це дозволяє користувачам завантажувати, створювати, редагувати та обмінюватися документами Microsoft Office безпосередньо в веб-браузері. Користувачі можуть створювати, переглядати та редагувати документи Word, Excel, PowerPoint і OneNote прямо в браузері. Безсумнівною перевагою сервісу є можливість запису файлів шляхом простого переміщення або використання веб-додатків. Присутній і віддалений доступ до комп'ютера, який працює під управлінням Windows.

Кількість доступних сховищ змінювалась кілька разів. Спочатку сервіс надавав 7 Гб пам'яті, і на 1 рік - ще 3 ГБ вільного сховища для студентів. Користувачі, які

підписалися на OneDrive до 22 квітня 2012 року, змогли прийняти обмежену пропозицію на 25 Гб безкоштовного оновлення. Сервіс побудований за допомогою технологій HTML5, а файли до 300 МБ можуть завантажуватися за допомогою перетягування в веб-браузері або до 10 Гб за допомогою настільної програми OneDrive для Microsoft Windows і OS X. З 23 вересня 2013 року, крім 7 Гб вільного місця для зберігання (або 25 Гб для користувачів, які мають право на безкоштовне оновлення), користувачі, які потребують більшого обсягу пам'яті, можуть вибрати один із чотирьох додаткових планів зберігання.

Користувачам у деяких регіонах може знадобитися певна платіжна картка або PayPal. Платіжний план зберігання поновлюється автоматично щороку, якщо компанія Microsoft або користувач не скасують цю послугу. Після повторного запуску в якості OneDrive було запроваджено щомісячні платіжні плани, а також можливість заробити до 5 Гб вільної пам'яті для переадресації нових користувачів на OneDrive (по 500 МБ кожна) та 3 ГБ, якщо користувачі активують автоматичне завантаження фотографій за допомогою мобільних додатки OneDrive на смартфонах. Абоненти домашніх планів Office 365 також отримують додатковий обсяг пам'яті для використання сервісу, з 20 Гб на користувача.

На відміну від своїх конкурентів, Dropbox та Google Drive, OneDrive не зберігає попередні версії файлів. Існує декілька версій файлів у форматах Microsoft Office, але не для інших файлів.

OneDrive реалізує "сміттєвий бак": файли, які користувач вибирає для видалення, зберігаються там протягом певного часу, не рахуючи їх частиною розподілу користувача, і можуть бути відновлені до тих пір, поки вони не будуть очищені від OneDrive.

Microsoft додав Office Online (відомий у той час як Office Web Apps) можливість OneDrive у своєму "Wave 4", що дозволяє користувачам завантажувати, створювати, редагувати та надсилати документи Word, Excel, PowerPoint та OneNote безпосередньо в веб-переглядачі. Крім того, Office Online дозволяє декільком користувачам одночасно співавторів документів Excel у веб-переглядачі та

співавторів документів OneNote з іншим веб-користувачем або настільною програмою. Користувачі також можуть переглядати історію версій документів Office, що зберігаються на OneDrive.

Microsoft випустила клієнтські програми OneDrive для Android, iOS, Windows 8, Windows 10, Windows 10 Mobile, Windows Phone Xbox 360, і Xbox One, які дозволяють користувачам переглядати та впорядковувати файли, що зберігаються на хмарному сховищі OneDrive. Крім того, Microsoft випустила настільні додатки для Microsoft Windows (Vista і пізніших версій) та OS X (10.7 Lion і пізніших версій), які дозволяють користувачам синхронізувати весь об'єм OneDrive з їх комп'ютерами для автономного доступу, а також між кількома комп'ютерами. Клієнт OneDrive для Windows дозволяє користувачам завантажувати вміст своїх ПК через веб-браузер, якщо користувач включив цю опцію, також користувачі macOS можуть завантажувати з ПК, але не навпаки. Версії Android, iOS та Windows Phone 8 також дозволяють автоматично завантажувати фотографії камери на OneDrive. Після ребрендингу в якості OneDrive додаток Xbox One також додав досягнення.

2.3. Технологія Blockchain

Мережу Blockchain можна уявити собі як безперервну послідовність залежних один від одного блоків, що містять як корисну інформацію, так і додаткові службові дані, які гарантують достовірність інформації, що зберігається (Рисунок 2.2).

Головною особливістю даної технології є розподіленість (децентралізація) – не існує єдиного центрального сховища даних, кожен користувач зберігає у себе повну копію всієї інформації [7]. Кожен екземпляр сховища синхронізується з іншими по заданих системою правилам. Другою важливою особливістю мережі Blockchain є її криптозахищеність. Під цим мається на увазі використання спеціалізованих алгоритмів хешування даних, які гарантують їх цілісність і незмінність. Це особливо важливо, якщо брати до уваги, що не існує еталона даних, кожен екземпляр є повноцінним і рівним іншому. Тому дуже важливо захищати дані від внесення змін.

При створенні блоку даних виконується хешування введених даних. Потім виконується хешування всіх збережених даних блоку, після чого цей результат записується в блок [8]. Даний підхід дозволяє гарантувати збереження даних, тому що при зміні будь-якого з полів блоку хоча б на один символ хеш функція поверне зовсім інший результат, і система легко зможе це визначити, виконавши повторне хешування блоку і порівнявши з збереженим хешем. При цьому кожен блок залежить від хешу попереднього блоку, в результаті чого при зміні будь-якого блоку в ланцюжку, весь ланцюжок стає некоректною (Рисунок 2.2).

Хешування – це процес перетворення вхідних даних довільної довжини в значення певного формату, шляхом перетворення за заданим алгоритмом. Основний особливість хеш-функції є можливість легкого перетворення вхідних даних в хеш, і неможливістю однозначного відновлення вихідних даних з хешу [9].

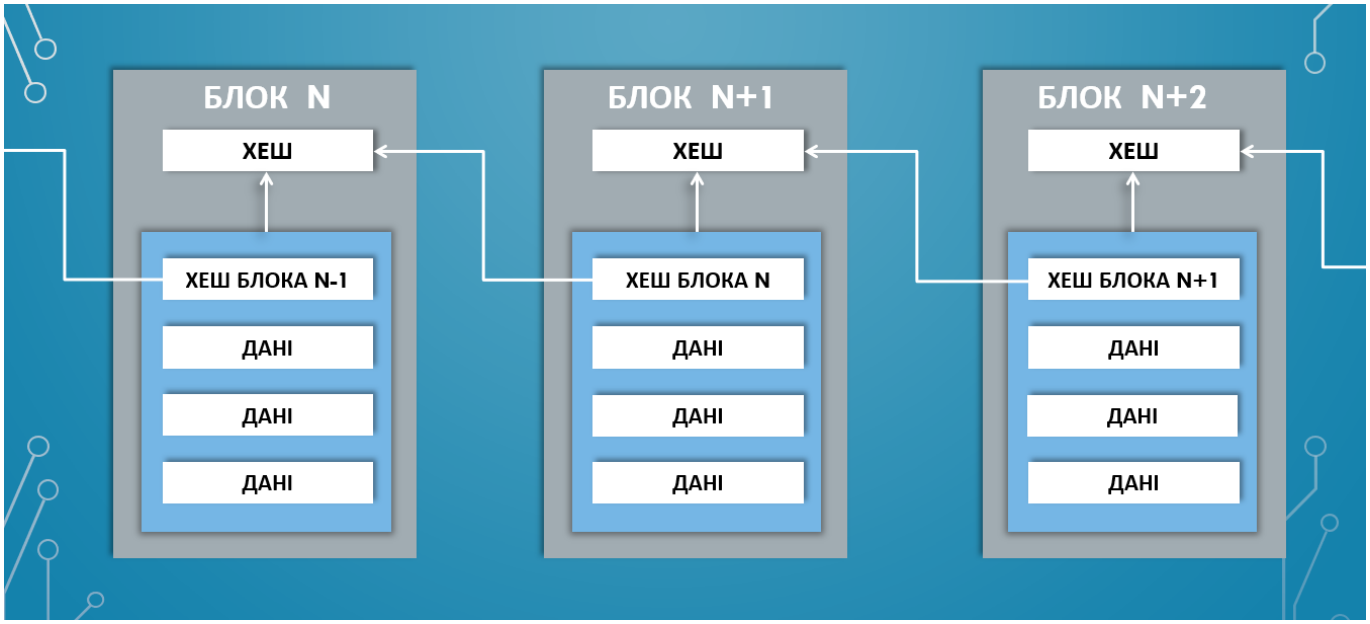

Рисунок 2.2 – Схема блоків в мережі Blockchain

Розглянемо приклад найелементарнішої хеш-функції, це підсумовування всіх цифр числа. Наприклад, якщо на вхід ми отримаємо число 73, то хешем даного числа буде 7 + 3 = 10. Це дуже проста операція, яка дозволяє отримати хеш. Але ось зворотне перетворення однозначно виконати неможливо, так як існує велика кількість числа, що дають такий же результат: 64, 37, 181, 11116 і так далі [10].

3. ЗАСОБИ РОЗРОБКИ

Основним середовищем розробки було середовище Visual Studio 2017. Програмний продукт розроблено по технології ASP .NET Core, для створення зовнішнього вигляду та інтерактивності веб-сторінок використовувався фреймворк Bootstrap.

3.1. Середовище розробки Microsoft Visual Studio 2017

Microsoft Visual Studio – це програмне середовище з розробки додатків для ОС Windows, як консольних, так і з графічним інтерфейсом.

У комплект входять наступні основні компоненти:

– Visual Basic.NET – для розробки додатків на VisualBasic;
– Visual C++ – на традиційній мові C++;
– Visual C# – на мові C# (Microsoft);
– Visual F# – на мові F# (Microsoft Developer Division).

Функціональна структура середовища включає в себе:

– редактор вихідного коду, який включає безліч додаткових функцій, як автодоповнення IntelliSense, рефракторінг коду і т. д.;
– відладчик коду;
– редактор форм, призначений для спрощеного конструювання графічних інтерфейсів;
– веб-редактор;
– дизайнер класів;
– дизайнер схем баз даних.

Visual Studio також дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (Subversion і VisualSourceSafe), додавання нових наборів інструментів (для редагування і

візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів процесу розробки програмного забезпечення) [11].

Нижче перераховані основні переваги середовища розробки Visual Studio.

Вбудований Web-сервер. Для обслуговування Web-додатків ASP.NET необхідний Web-сервер, який буде очікувати Web-запити і обробляти відповідні сторінки. Наявність в Visual Studio інтегрованого Web-сервера дозволяє запускати Web-сайт прямо з середовища проектування, а також підвищує безпеку, виключаючи ймовірність отримання доступу до тестового Web-сайту з якого-небудь зовнішнього комп'ютера, оскільки тестовий сервер може приймати з'єднання лише з локального комп'ютера.

Visual Studio дозволяє писати код своєю рідною мовою чи будь-якими іншими бажаними мовами, використовуючи весь час один і той же інтерфейс (IDE). Більш того, Visual Studio також ще дозволяє створювати Web-сторінки на різних мовах, але поміщати їх в один і той же Web-додаток. Єдиним обмеженням є те, що в кожній Web-сторінці можна використовувати тільки якусь одну мову (очевидно, що в іншому випадку проблем при компіляції було б просто не уникнути).

Для створення більшості додатків потрібно пристойну кількість стандартного стереотипного коду, і Web-сторінки ASP. NET тому не виключення. Наприклад, додавання Web-елемента управління, приєднання обробників подій і коригування форматування вимагає установки в розмітці сторінки ряду деталей. У Visual Studio такі деталі встановлюються автоматично.

Більш висока швидкість розробки. Багато з функціональних можливостей Visual Studio спрямовані на те, щоб допомагати розробнику робити свою роботу якомога швидше. Зручні функції, на зразок функції IntelliSense (яка вміє перехоплювати помилки і пропонувати правильні варіанти), функції пошуку і заміни (яка дозволяє відшукувати ключові слова як в одному файлі, так і в усьому проекті) і функції автоматичного додавання і видалення коментарів (яка може тимчасово приховувати блоки коду), дозволяють розробнику працювати швидко і ефективно.

Можливості налагодження. Пропоновані в Visual Studio інструменти налагодження є найкращим засобом для відстеження помилок і діагностування дивної

поведінки. Розробник може виконувати свій код по рядку за раз, встановлювати інтелектуальні точки переривання, при бажанні зберігаючи їх для використання в майбутньому, і в будь-який час переглядати поточну інформацію з пам'яті.

Як недолік можна відзначити неможливість відладчика (Microsoft Visual Studio Debugger) відстежувати в коді режиму ядра. Налагодження в Windows в режимі ядра в загальному випадку виконується при використанні WinDbg, KD або SoftICE.

Під час розробки даного програмного забезпечення було використано версію редакції Enterprise. До складу цієї версії входить модуль візуального проектування GUI-інтерфейсу Swing UI Designer, XML-редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з Eclipse. Доступні засоби інтеграції з системами відстеження помилок JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack, Lighthouse.

Схема змін та нововведень у Visual Studio 2017 (Рисунок 3.1).

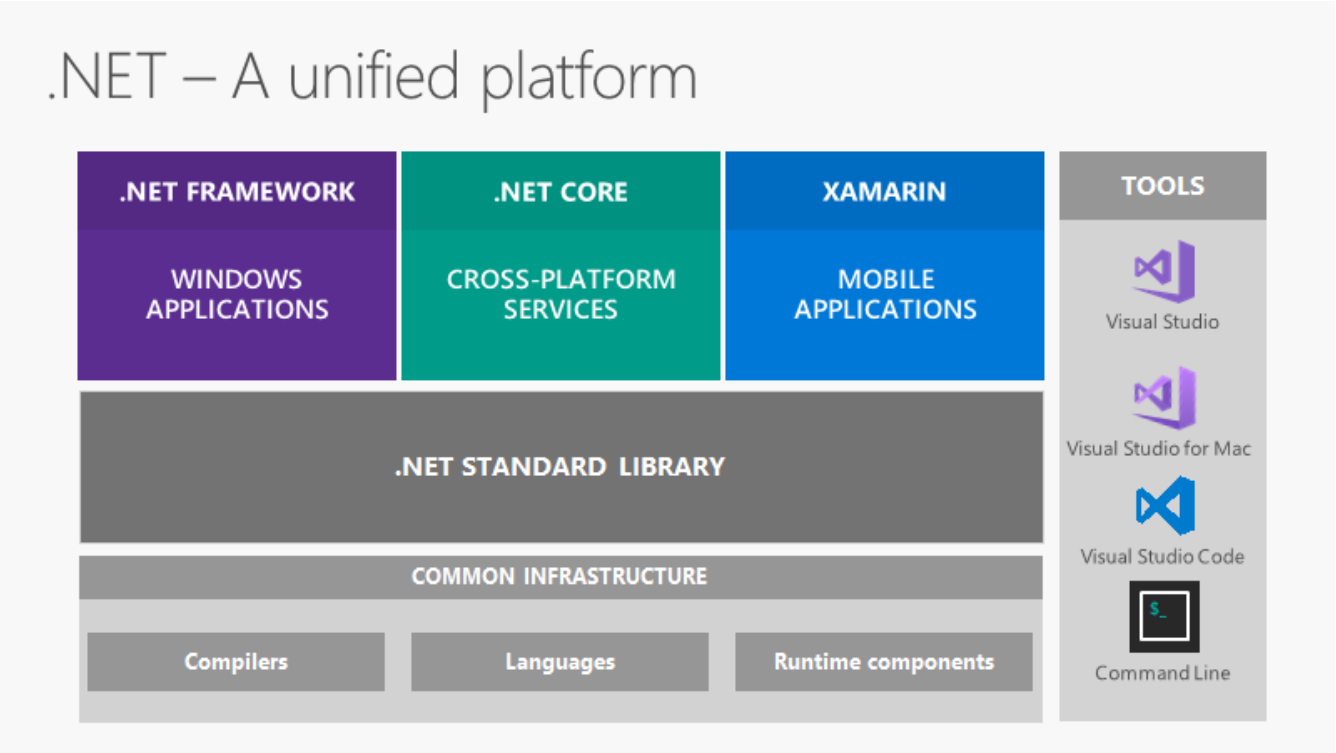

Рисунок 3.1 – Зміни та нововведення у Visual Studio 2017

Особливості Visual Studio 2017:

– покращена інтеграція з хмарною технологією Azur. Розробки Microsoft в цьому напрямку дозволяють полегшити створення, налагодження, розміщення і публікацію ваших додатків в хмарі Azure прямо з IDE, надаючи до того ж вбудовані інструменти для роботи цими додатками, а також з Docker-контейнерами, .NET Core додатками і так далі;
– розробка для мобільних пристроїв. Розробники отримали поліпшені інструменти налагодження і профілювання, інструменти генерації модульних тестів. Також з'явилася можливість створювати кросплатформені додатки;
– офіційно доступна нова версія Visual studio Team Foundation Server 2017. У цей випуск on-premise платформи для організації спільної роботи команд включили давно очікувані можливості, наприклад, нові шаблони процесів, поліпшене керування доступом до репозиторіїв, pull-реквестамі і багато іншого;
– розробники отримують доступ до додаткових сервісів для оптимізації і створення DevOps циклу всередині своєї організації, таким як хмарний CI-сервер, інструменти навантажувального тестування в хмарі і навіть персонального DevOps навчання;
– випуск нового інструментарію, доступного .NET Core в складі Visual Studio 2017;
– у CLI додалися додаткові команди і можливість вибору власних шаблонів проекту. Також був анонсований приклад реалізації мікросервісної архітектури, який ви можете знайти в репозиторії GitHub;
– нові можливості торкнулися структури проекту, заснованої на .csproj, що забезпечує сумісність з build-системами для .NET, заснованими на MSBuild. Додатково, формат.csproj значно спрощує розробникам можливості редагування файли для оголошення залежностей, target-платформ і властивостей проекту.

3.2. Мова програмування C#

Мова програмування C# – об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

C# можна використовувати для створення клієнтських додатків Windows, XML-веб-служб, розподілених компонентів, додатків клієнт-сервер, додатків баз даних і т. д. Visual C# надає вдосконалений редактор коду, зручні конструктори користувальницького інтерфейсу, інтегрований відладчик і багато інших засобів, щоб спростити розробку додатків на мові C# і платформі .NET Framework [12].

Синтаксис C# дуже багатий, але при цьому простий і зручний у вивченні. Характерні фігурні дужки C # миттєво впізнаються всіма, хто знайомий з C, C++ або Java. Розробники, які знають будь-який з цих мов, зазвичай дуже швидко починають ефективно працювати в C#. Синтаксис C# спрощує багато складності C++, але при цьому надає відсутні в Java потужні функції, наприклад обнуляє типи значень, перерахування, делегати, лямбда-вирази і прямий доступ до пам'яті. C# підтримує універсальні методи і типи, які забезпечують більш високий рівень безпеки і продуктивності, а також ітератори, що дозволяють визначати в класах колекцій власну поведінку ітерації, яке може легко застосувати в клієнтському коді. Вирази LINQ створюють дуже зручну мовну конструкцію для строго типізованих запитів.

C# є об'єктно-орієнтованою мовою, а значить підтримує інкапсуляцію, успадкування і поліморфізм. Всі змінні і методи, включаючи метод Main, який представляє собою точку входу в додаток, инкапсулируются в визначення класів. Клас успадковується безпосередньо з одного батьківського класу, але може реалізовувати будь-яке число інтерфейсів. Методи, які скасовують віртуальні методи батьківського класу, повинні містити ключове слово override, щоб виключити випадковий перевизначення. У мові C# структура схожа на полегшений клас: це тип, що розподіляється в стеці, який реалізує інтерфейси, але не підтримує спадкування.

Крім цих основних принципів об'єктно-орієнтованого програмування, C# пропонує ряд інноваційних мовних конструкцій, що спрощують розробку програмних компонентів:

– інкапсульовані сигнатури методів, іменовані делегатами, які дозволяють реалізувати типобезпечне повідомлення про події;
– властивості, що виконують функцію акцессоров для закритих змінних-членів;
– атрибути, що надають декларативні метадані про типи під час виконання;
– внутрішньорядкові коментарі для XML-документації;
– LINQ для створення запитів до різних джерел даних.

Для взаємодії з іншим програмним забезпеченням Windows, наприклад з об'єктом COM або власними бібліотеками DLL Win32, ви можете застосувати процес C#, відомий як "Взаємодія". Взаємодія дозволяє програмам на C # робити практично все, що можливо в додатку машинного коду C++. C# підтримує навіть покажчики і поняття "небезпечного" коду для тих випадків, в яких критично важливий прямий доступ до пам'яті.

Процес побудови в C# простіше в порівнянні з C або C++, але більш гнучкий, ніж в Java. Окремі файли заголовка не використовуються, і немає необхідності оголошувати методи і типи в певному порядку. Вихідний файл C # може визначити будь-яке число класів, структур, інтерфейсів і подій.

Програми C# виконуються на платформі .NET Framework, вбудованому компоненті Windows, яка включає віртуальну систему виконання, звану підтримкою загальномовного середовища виконання (CLR), і уніфікований набір бібліотек класів. Середовище CLR корпорації Майкрософт представляє собою комерційну реалізацію міжнародного стандарту Common Language Infrastructure (CLI), який служить основою для створення середовищ виконання і розробки, що дозволяють спільно використовувати різні мови і бібліотеки.

Вихідний код, написаний на мові C# компілюється в проміжну мову (IL), який відповідає специфікаціям CLI. Код на мові IL і ресурси, в тому числі точкові малюнки і рядки, зберігаються на диск у вигляді виконуваного файлу (зазвичай з розширенням

.exe або .dll). Такий файл називається збіркою. Збірка містить маніфест з інформацією про типи, версії, вимог безпеки, мовою і регіональних параметрах для цієї збірки.

При виконанні програми C# середовище CLR завантажує збірку і виконує різні дії в залежності від відомостей, збережених в маніфесті. Якщо виконуються всі вимоги безпеки, середовище CLR виконує JIT-компіляцію з коду на мові IL в інструкції машинної мови. Також середовище CLR виконує інші операції, наприклад автоматичну збірку сміття, обробку винятків і управління ресурсами. Код, що виконується середовищем CLR, іноді називають "керованим кодом", щоб підкреслити відмінності цього підходу від "некерованого коду", який відразу компілюється в машинну мову для певної системи [13]. На Рисунку 3.2 показані зв'язки між файлами вихідного коду C#, бібліотеками класів .NET Framework, збірками і середовищем CLR, існуючі під час компіляції і під час виконання.

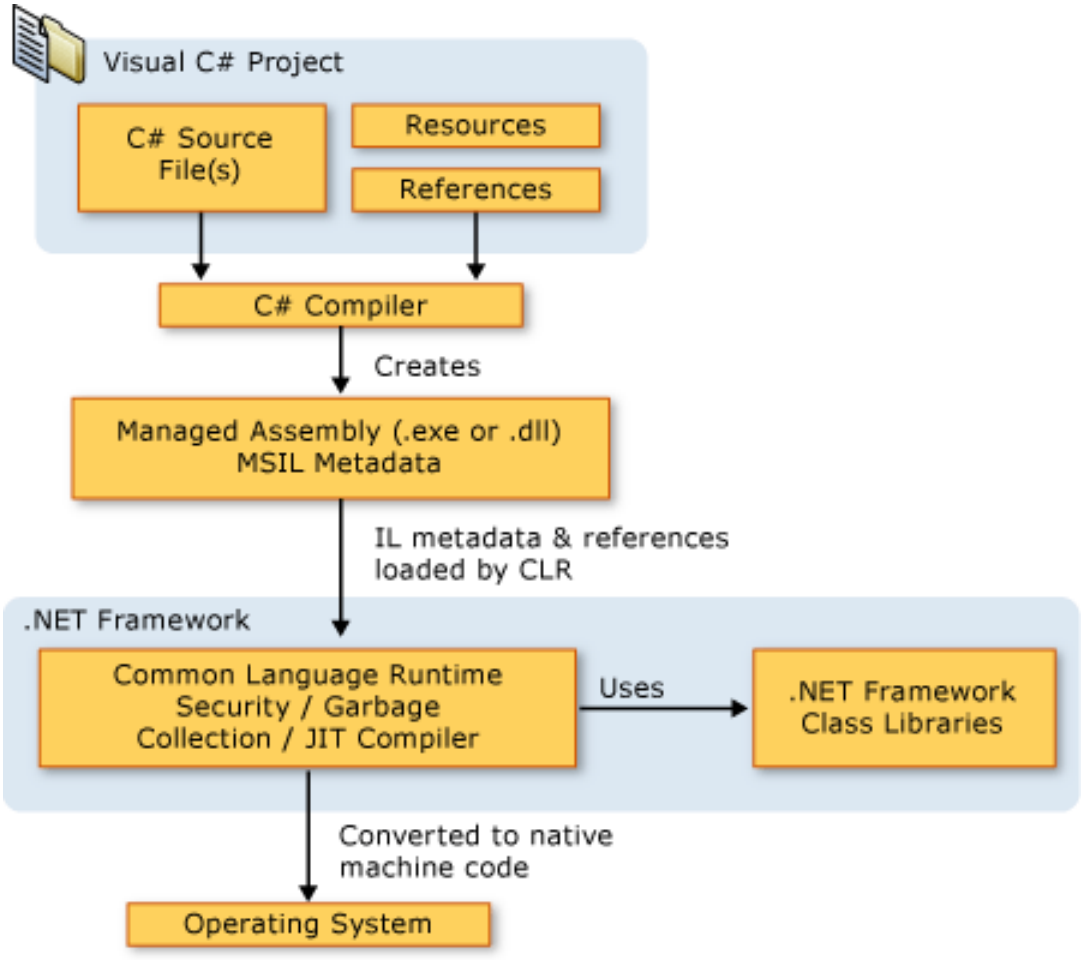

Рисунок 3.2 – Схема зв’язків між файлами вихідного коду C#

Взаємодія між мовами є ключовою особливістю платформи .NET Framework. Створюваний компілятором C# код IL відповідає специфікації загальних типів (CTS). Це означає, що цей код IL може успішно взаємодіяти з кодом, створеним з Visual Basic і Visual C++ для платформи .NET або з будь-якого іншого CTS-сумісного мови, яких існує вже більше 20. Одна збірка може містити кілька модулів, написаних на різних мовах .NET, і всі типи можуть посилатися один на одного, як якщо б вони були написані на одній мові.

Крім служб в середовищі виконання, платформа .NET Framework також включає велику бібліотеку з більш ніж 4000 класів, організованих по просторах імен, які надають різноманітні корисні функції для всіх операцій: від введення і виведення файлів до управління рядками при аналізі XML і елементів управління Windows Forms . Типове додаток C# широко використовує бібліотеки класів .NET Framework для стандартних завдань по підключенню [14].

3.3. Платформа ASP. NET Core

Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів [15].

З одного боку, ASP.NET Core є продовженням розвитку платформи ASP.NET. Але з іншого боку, це не просто черговий реліз. Вихід ASP.NET Core фактично означає революцію всієї платформи, її якісна зміна.

Розробка над платформою почалася ще в 2014 році. Тоді платформа умовно називалася ASP.NET vNext. У червні 2016 року вийшов перший реліз платформи. А в травні 2018 року побачила версія ASP.NET Core 2.1, яка власне і охоплена в поточному керівництві.

ASP.NET Core тепер повністю є opensource-фреймворком. Всі вихідні файли фреймворку доступні на GitHub.

ASP.NET Core може працювати поверх крос-платформної середовища .NET Core, яка може бути розгорнута на основних популярних операційних системах: Windows, Mac OS X, Linux. І таким чином, за допомогою ASP.NET Core ми можемо створювати крос-платформні додатки. І хоча Windows як середовище для розробки і розгортання програми досі превалює, але тепер вже ми не обмежені тільки цією операційною системою. Тобто ми можемо запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. А для розгортання веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

Хоча ASP.NET Core переважно націлене на використання .NET Core, але фреймворк також може працювати і з повною версією фреймворка .NET.

Завдяки модульності фреймворка всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний менеджер Nuget. Крім того, на відміну від попередніх версій платформи немає необхідності використовувати бібліотеку System.Web.dll.

ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API і Web Pages. У попередніх версії платформи дані технології реалізувалися окремо і тому містили багато дублюючої функціональності. Зараз же вони об'єднані в одну програмну модель ASP.NET Core MVC. А Web Forms повністю пішли в минуле.

Крім об'єднання вищезазначених технологій в одну модель в MVC був доданий ряд додаткових функцій.

Однією з таких функцій є тег-хелпери (tag helper), які дозволяють більш органічно поєднувати синтаксис html з кодом C #.

ASP.NET Core характеризується розширюваністю. Фреймворк побудований з набору щодо незалежних компонентів. І ми можемо або використовувати вбудовану реалізацію цих компонентів, або розширити їх за допомогою механізму спадкування, або зовсім створити і застосовувати свої компоненти зі своїм функціоналом.

Також було спрощено управління залежностями і конфігурація проекту. Фреймворк тепер має свій легкий контейнер для впровадження залежностей, і більше

немає необхідності застосовувати сторонні контейнери, такі як Autofac, Ninject. Хоча при бажанні їх також можна продовжувати використовувати.

В якості інструментарію розробки ми можемо використовувати останні випуски Visual Studio, починаючи з версії Visual Studio 2015. Крім того, ми можемо створювати додатки в середовищі Visual Studio Code, яка є крос-платформної і може працювати як на Windows, так і на Mac OS X і Linux.

Для обробки запитів тепер використовується новий конвеєр HTTP, який заснований на компонентах Katana і специфікації OWIN. А його модульність дозволяє легко додати свої власні компоненти.

Якщо підсумувати, то можна виділити наступні ключові відмінності ASP.NET Core від попередніх версій ASP.NET:

– новий легкий і модульний конвеєр HTTP-запитів;

– можливість розгортати додаток як на IIS, так і в рамках свого власного процесу;

– використання платформи .NET Core і її функціональності;

– поширення пакетів платформи через NuGet;

– інтегрована підтримка для створення та використання пакетів NuGet;

– єдиний стек веб-розробки, що поєднує Web UI і Web API;

– конфігурація для спрощеного використання в хмарі;

– вбудована підтримка для впровадження залежностей;

– можливість розширення;

– кросплатформеність: можливість розробки і розгортання додатків ASP.NET на Windows, Mac і Linux;

– розвиток як open source, відкритість до змін.

Ці та інші особливості і можливості стали основою для нової моделі програмування.

Web API – спосіб побудови програми ASP.NET, що спеціально створений для роботи в стилі REST (Representation State Transfer або "передача стану") [16]. REST-архітектура передбачає застосування таких методів або типів запитів HTTP для взаємодії з сервером:

– GET (отримати);
– POST (відправити);
– PUT (помістити);
– DELETE (видалити).

Даний набір методів служить основою базових CRUD-операцій (Create, Read, Update, Delete). REST-стиль особливо зручний при створенні будь-якого Single Page Application (SPA), які нерідко використовують спеціальні Javascript-фреймворки, наприклад, Angular, React або Vue. По суті, Web API представляє собою веб-службу, до якої можуть звертатися інші додатки. Причому ці програми можуть представляти будь-яку технологію і платформу - це можуть бути веб-додатки, мобільні або настільні клієнти.

3.4. Фреймворк Bootstrap

Bootstrap – це CSS фреймворк, який спочатку створювався для внутрішнього використання компанією «Twitter» з робочою назвою «Twitter Blueprint», але в підсумку був опублікований у відкритий доступ і став хорошим набором інструментів для front-end розробки під назвою «Bootstrap» [17]. Офіційний сайт фреймворка знаходиться за адресою getbootstrap.com.

Хоча Bootstrap і називають CSS фреймворком, але це не зовсім вірно. На мій погляд, правильніше його називати WEB фреймворком, так як він містить готові CSS, HTML і JavaScript компоненти, а третя версія має власний іконочние шрифт.

Шрифт містить більше 250 іконок. Кількість іконок, звичайно, не таке велике, як у Font Awesome, але все базові іконки присутні. З четвертої версії фреймворк відмовився від власного іконочние шрифту на користь використання сторонніх бібліотек, які необхідні користувачу для конкретного проекту.

При верстці адаптивного класичного макета: шапка сайту (header), основна частина (content), бічна колонка (sidebar) і підвал сайту (footer), для коректного відображення нам потрібно розрахувати ширину у відсотках кожного елемента і

привласнити обтікання. Якщо з шапкою і футером все зрозуміло, в більшості випадків ширина буде 100%, то для основної частини контенту і бічний колонки може бути 70/30 або 85/25, але при зменшенні екрану нас це не влаштує, потрібно буде робити по 100% і скидати обтікання.

Ось для таких цілей і потрібна сітка Bootstrap. Просто задаються класи для блоків, які вказують, яку ширину повинен займати елемент і як він буде відображатися на різних пристроях. Сітка функціонує як таблиця, в якій є свої ряди і стовпці, максимальна кількість стовпців 12.

Сітку можна робити всередині іншої сітки скільки завгодно. Якщо робити все блоки сайту з використанням сітки, то самостійно писати медіа запити для їх адаптивності взагалі не доведеться.

Крім сітки існує величезна кількість всіляких компонентів: навігаційні меню, форми, таблиці, модальні вікна, вкладки, оповіщення, спливаючі підказки і т.д.

Переваги фреймворка Bootstrap:

- висока швидкість розробки макетів сторінок сайту. Bootstrap містить величезний набір готових рішень і елементів;
- кросбраузерність і адаптивність сайту. Всі елементи фреймворка адаптивні під всі пристрої і коректно відображаються у всіх сучасних браузерах;
- легкість у використанні. Навіть людина, що має базові знання про HTML і CSS, може вільно створювати web-сторінки з використанням фреймворку;
- простота в навчанні. У Bootstrap дуже хороша документація з великою кількістю прикладів готового коду.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Основна розробка програмного продукту поділялася на 2 частини: побудова візуального інтерфейсу для користувача та реалізації логіки збереження та обробки даних.

4.1. Архітектура програмної системи

Програмний продукт спроектований та розроблений згідно шаблону проектування MVC [18] (Рисунок 4.1).

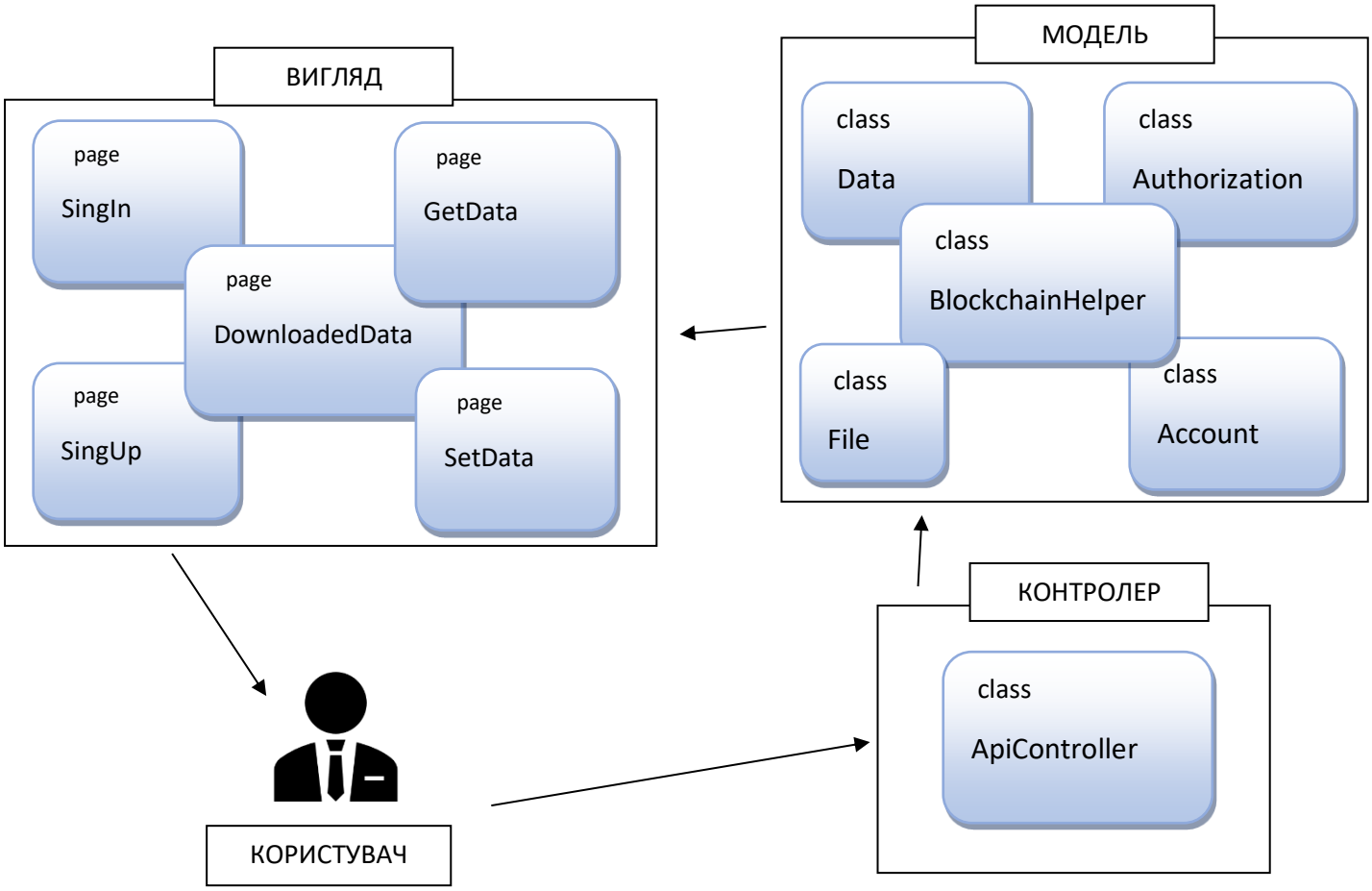

Рисунок 4.1 – Схема роботи програмного продукту

Модель системи складається з класів Data, BlockchainHelper, File, Account, Authorization які місять в собі моделі та логіку обробки даних.

Контролер ApiController відповідальний за обробку усіх HTTP-запитів, які є дозволеними для користувача. Всі можливі запити подано у Таблиці 4.1.

Таблиця 4.1 – Обробка HTTP-запитів у контролері системи.

Шлях	HTTP-метод доступу	Аргументи методу контролера
SingIn	GET/POST	/Authorization authorizationData
SingUp	GET/POST	/Authorization authorizationData
SetData	GET/POST	/IFormFile file
GetData	GET/POST	/IFormFile file
DownloadedData	GET	

Вигляд системи складається з сторінок:

– SingIn, сторінка для авторизації користувача в мережі;
– SinUp, сторінка для реєстрації користувача в мережі;
– SetData, сторінка для завантаження даних в мережу;
– GetData, сторінка для отримання даних з мережі;
– DownloadedData, сторінка для перегляду завантажених даних.

Для створення кожного з модулів були використані технології, що не залежать від конкретної операційної системи чи апаратної платформи, тому вони можуть виконуватися для практично будь-якої з них.

4.2. Користувацький інтерфейс програмної системи

Інтерфейс системи складається з чіткого набору директорій, в кожній з яких знаходяться файли, що виконують подібні завдання та мають схожу структуру:

– Views (шаблони сторінок);

– App_Start (інформація про розгортання системи на сервері);

– bin (динамічні бібліотеки, створені при компіляції проекту);

– Models (класи бізнес-логіки);

– Controllers (контролери).

Шаблони сторінок написані на мовах HTML та C# з використанням фремворка Bootstrap та утворюють безпосередній інтерфейс для кінцевого користувача, приймаючи вхідні дані, відображаючи їх та сприймаючи відповіді.

На Рисунку 4.2 можна ознайомитись з усіма діями користувача в системі.

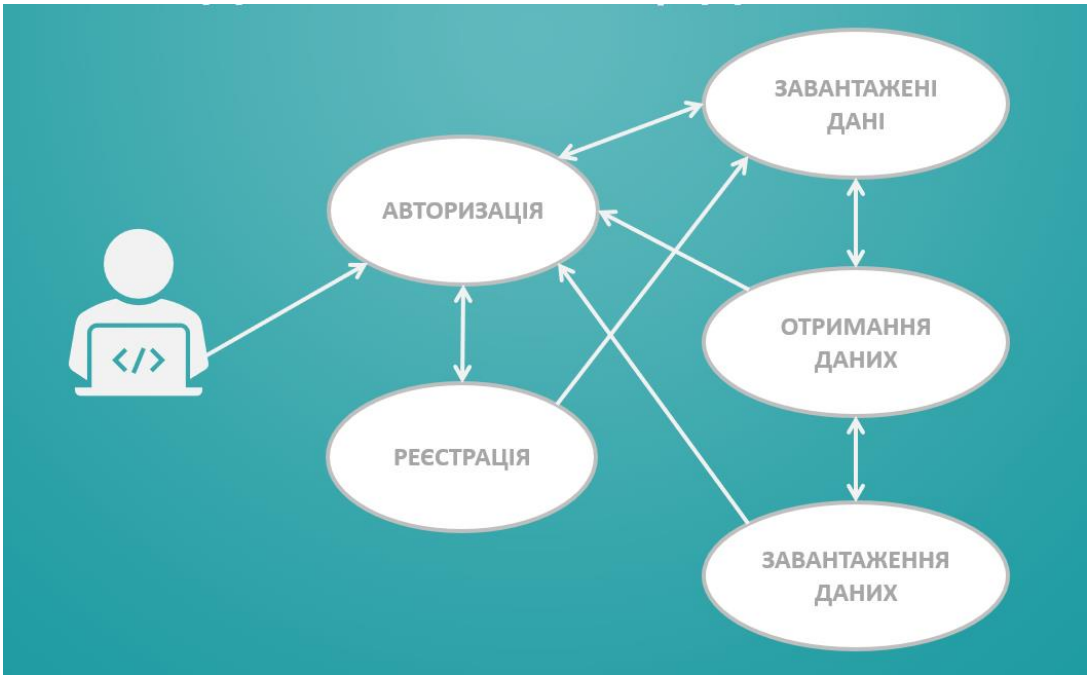

Рисунок 4.2.1 – Діаграма прецедентів

В результаті, інтерфейс створеної системи має достатньо просту структуру, яка відображає функціонал, доступний кінцевому користувачу.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА ПРОГРАМНИМ ПРОДУКТОМ

У даному розділі описано системні вимоги до ПК для забезпечення стабільної та коректної роботи розробленої системи, інструкцію користувача для її встановлення, а також детальну методику роботи користувача із програмною системою.

5.1. Системні вимоги

Для коректної роботи програмного продукту персональний комп'ютер має відповідати таким мінімальним системним та апаратним вимогам:

- встановлена операційна система Windows 7 32 або 64-розрядна;
- процесор із тактовою частотою 2 ГГц або швидший – 32-розрядний (x86) або 64-розрядний (x64);
- оперативна пам'ять 2 гігабайт (ГБ) (для 32-розрядної версії) або 4 ГБ (для 64-розрядної версії).

5.2. Інсталяція системи

Вимоги до попередньо встановленого на машину програмного забезпечення:

- встановлений .Net Framework версії 4.0 і вище;
- встановлене середовище розробки Microsoft Visual Studio 2017 та додаткові розширення для публікування програмних продуктів;
- встановлений браузер.

5.3. Сценарій роботи користувача з системою

При запуску програмного застосунку користувач спочатку бачить головне вікно в якому йому потрібно пройти процес авторизації (Рисунок 5.1).

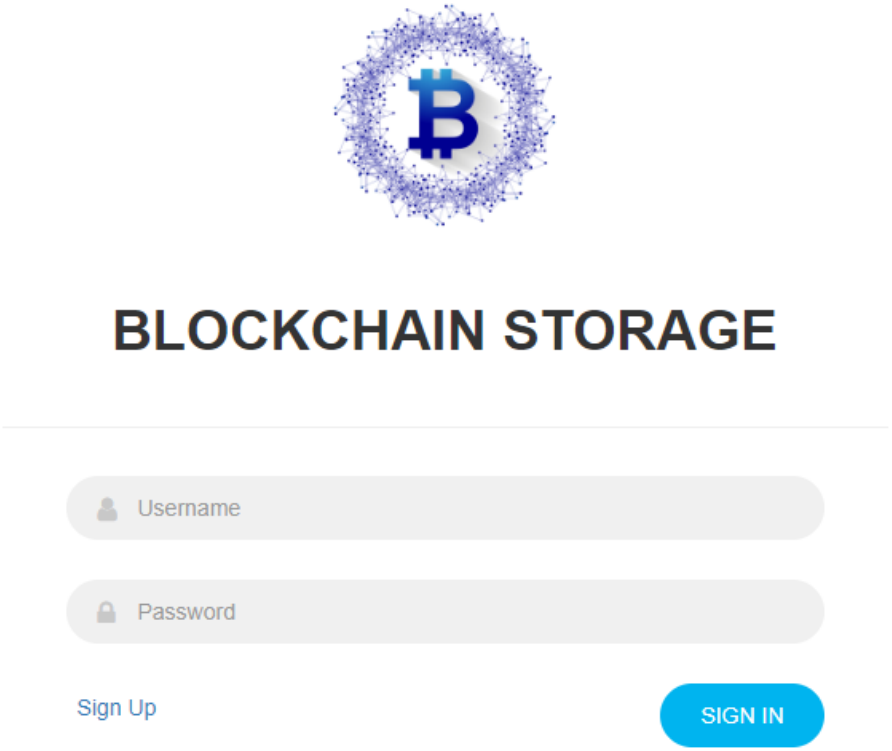

Рисунок 5.1 – Вікно авторизації користувача

Якщо користувач не зареєстрований в мережі, він може пройти процес реєстрації натиснувши відповідну клавішу (Рисунок 5.2).

Рисунок 5.2 – Вікно реєстрації користувача

Так як було використано шаблон MVC, то дані відображалися на сторінці за допомогою моделей, що значно спростило динамічну зміну інтерфейсу.

Після проходження авторизації користувач потрапляє на вікно власного кабінету в якому він відразу може переглянути свої завантажені дані в мережі (Рисунок 5.3).

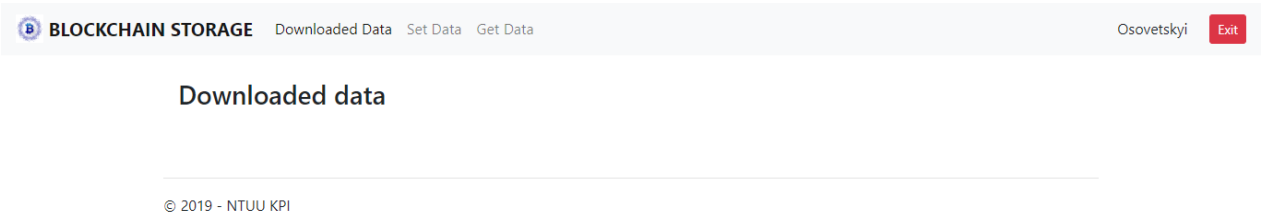

Рисунок 5.3 – Завантажені дані в мережу

В даному випадку у завантажені дані відсутні, так як користувач тільки пройшов реєстрацію в мережі.

Для завантаження даних в мережу користувачу необхідно перейти на відповідний екран вибравши відповідний пункт (Рисунок 5.4).

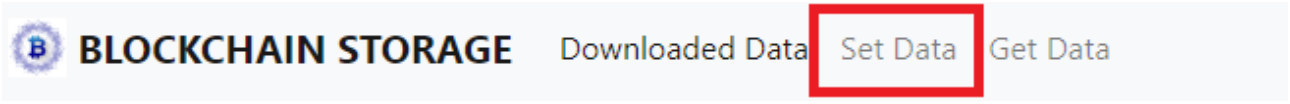

Рисунок 5.4 – Перехід на екран завантаження даних в мережу

Для завантаження даних у мережу користувач має вибрати файл, який він бажає додати та ввести для нього ключ, по якому даний файл буде збережено в мережі (Рисунок 5.5).

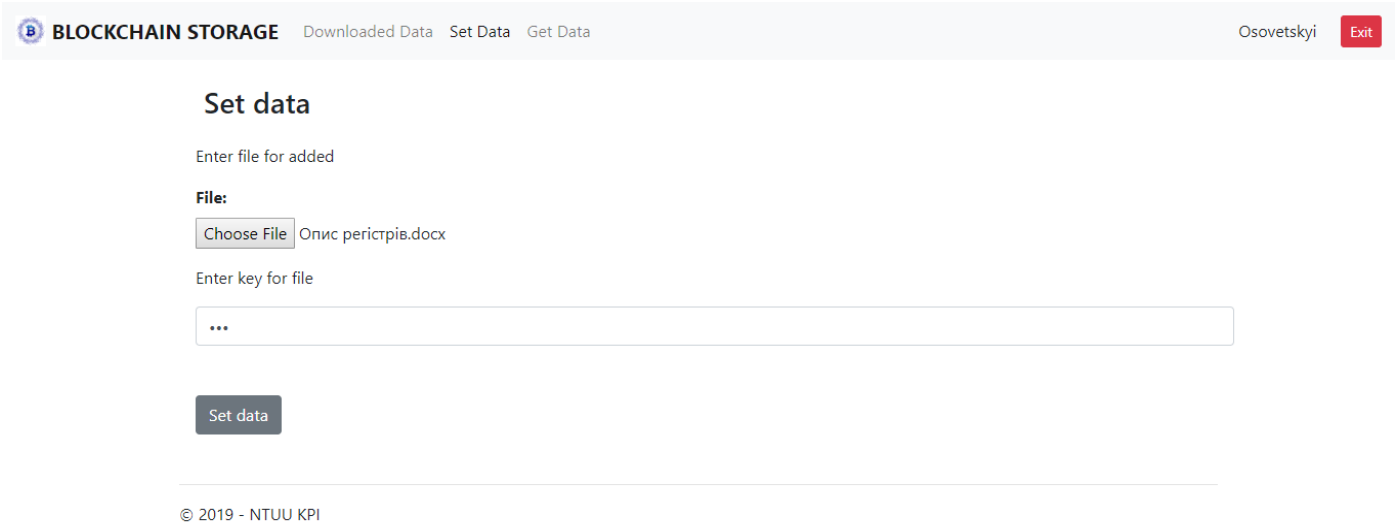

Рисунок 5.5 – Завантаження файлу в мережу

Якщо всі дані введені коректно, користувач отримає повідомлення про додавання файлу (Рисунок 5.6).

Рисунок 5.6 – Повідомлення про успішне завантаження файлу

Після успішного завантаження файлу користувача буде перенаправлено на вікно перегляду завантажених даних, в якому він уже зможе знайти даний файл (Рисунок 5.7).

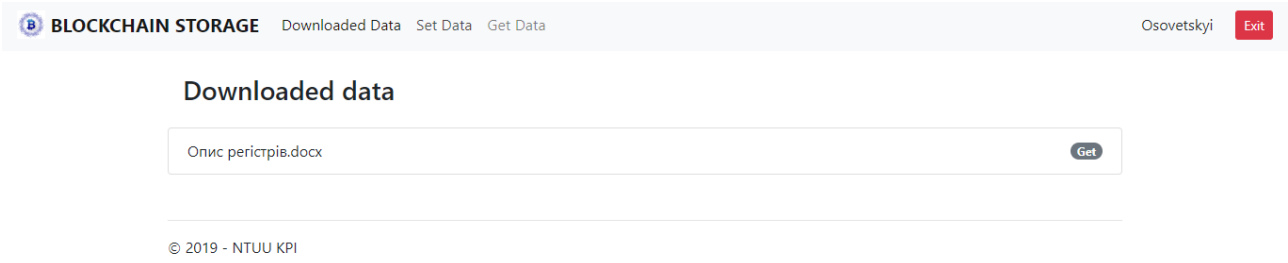

Рисунок 5.7 – Доданий файл

Для отримання даних користувачу потрібно перейти на вікно завантаження даних вибравши один із пунктів (Рисунок 5.8).

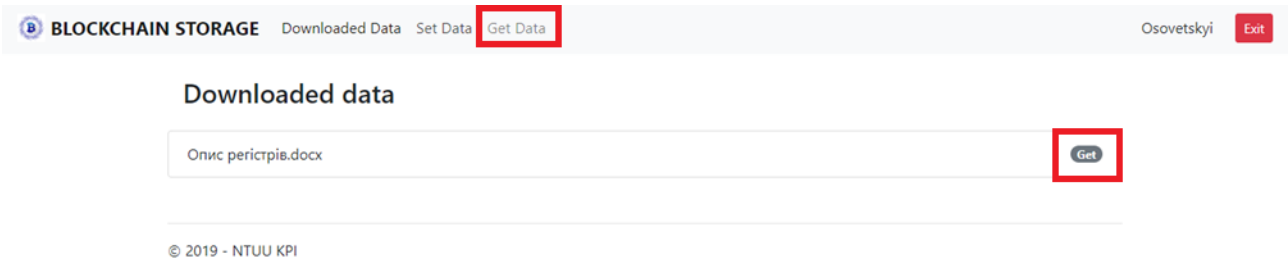

Рисунок 5.8 – Перехід на екран отримання даних

Для отримання завантажених даних, користувач має вказати шлях збереження файлу та вказати ключ відповідного файлу, який він хоче завантажити (Рисунок 5.9).

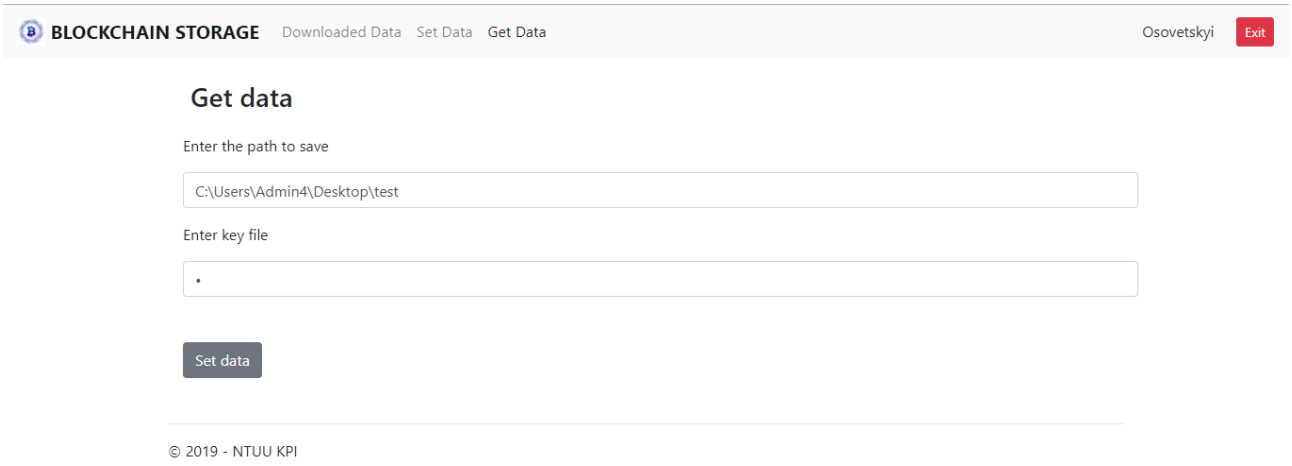

Рисунок 5.9 – Вікно отримання файлу з мережі

Якщо користувач введе неправильний ключ, файл не буде завантажено і користувач отримає відповідне повідомлення (Рисунок 5.10).

Рисунок 5.10 – Повідомлення про невдале отримання файлу

При правильно введених даних користувач отримає повідомлення про успішне завантаження файлу (Рисунок 5.11).

Рисунок 5.11 – Повідомлення про успішне отримання файлу

Після даного повідомлення користувач може перейти по вказаному шляху збереження файлу і переконатись в його успішному отриманні (Рисунок 5.12).

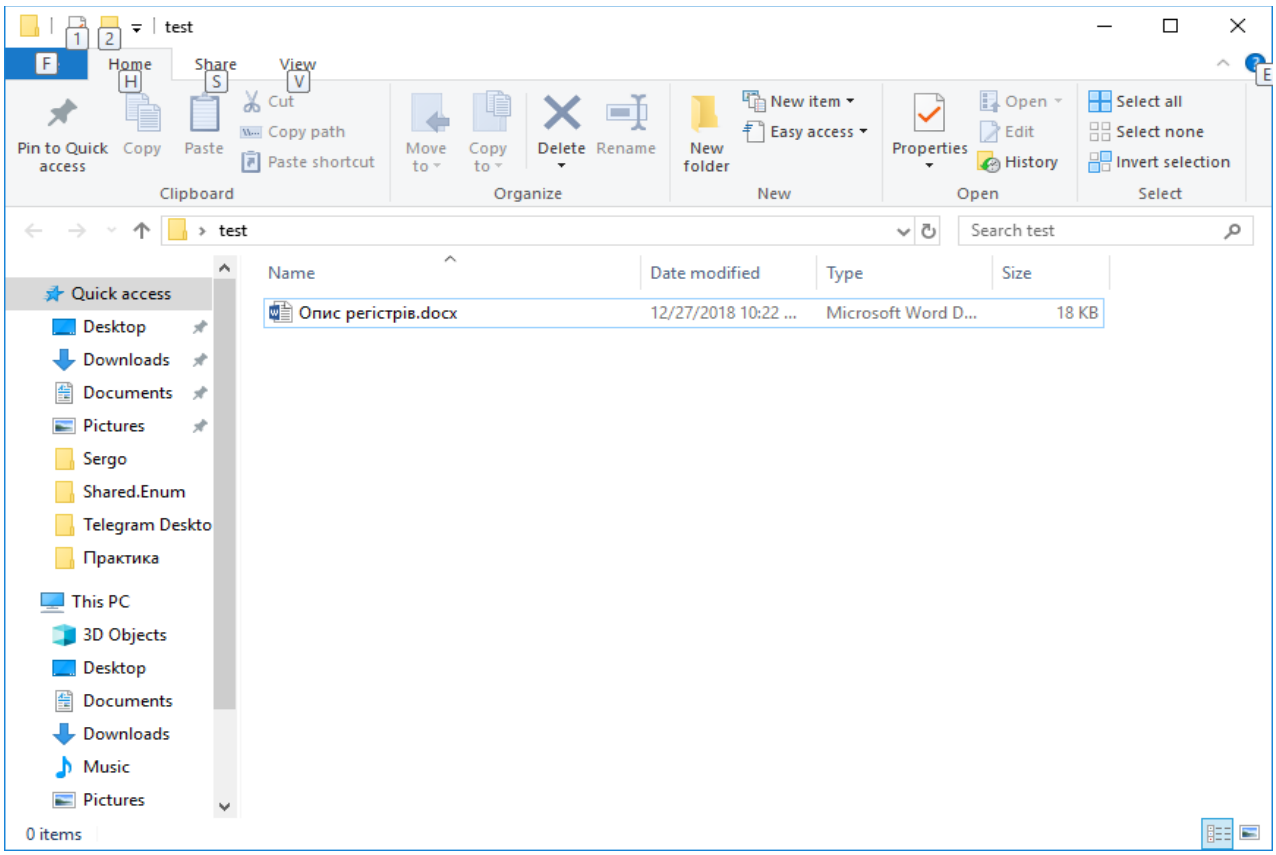

Рисунок 5.12 – Завантажений файл з мережі

Після завершення всіх операцій користувач має вийти з системи вибравши відповідний пункт (рисунок 5.13).

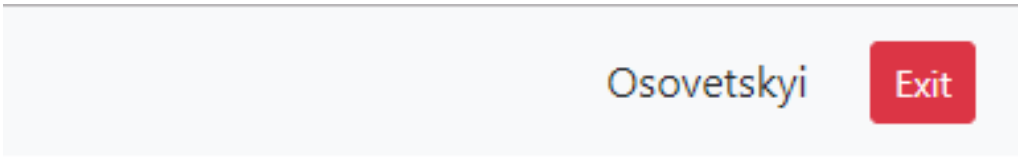

Рисунок 5.13 – Ідентифікатор користувача та клавіша виходу

Для повторного додавання чи отримання даних, користувачеві знову потрібно пройти авторизацію – такий функціонал дає більший рівень безпеки мережі.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено програмний продукт для безпечного та доступного маніпулювання власною інформацією.

Для досягнення поставленої мети було:

- здійснено аналіз існуючих варіантів рішень задач безпечного збереження інформації;
- проаналізовано літературні джерела на відповідну тематику;
- обрано засоби та інструменти розробки програмного продукту;
- обрано методи розробки;
- розроблено архітектуру програмного забезпечення
- здійснено програмну реалізацію продукту для збереження інформації з використанням обраних технологій програмування.

Розроблений програмний продукт включає в себе наступний функціонал:

- реєстрація користувача в мережі;
- авторизація користувача в мережі;
- перегляд завантаженої інформації;
- завантаження інформації в мережу;
- отримання завантаженої інформації.

В результаті процесу дипломної роботи, маємо програмний продукт, який можна використовувати в усіх сферах інформаційних технологій, де необхідне безпечне маніпулювання даними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ю. Родичев. Нормативная база и стандарты в области информационной безопасности, 2017. – 495 с.
2. А. Бабаш, Е. Баранова, Д. Ларин. Информационная безопасность. История защиты информации в России, 2015. – 321 с.
3. В. Бондарев. Введение в информационную безопасность автоматизированных систем, 2016. – 287 с.
4. С. Нестеров. Основы информационной безопасности, 2016. – 345 с.
5. M. Wittig, A. Wittig. Amazon Web Services in Action (1-st Edition), 2018. – 528 с.
6. Е. Шмидт. Как работает Google, 2015. – 384 с.
7. Н. Прасти. Блокчейн. Разработка приложений, 2016. – 256 с.
8. Л. Лелу. Блокчейн от А до Я. Все о технологии десятилетия, 2008. – 423 с.
9. А. Астрель. Книга шифрів. Таємна історія шифрів і їх розшифровки, 2007. – 446 с.
10. Ф. Бауэр. Расшифрованные секреты. Методы и принципы криптологии, 2007. – 550 с.
11. И. Гриффитс. Programming C# 5.0: Building Windows 8, Web, and Desktop Applications for the .NET 4.5 Framework / Ian Griffiths – Программирование на C# 5.0, 2015. – 895 с.
12. Г. Шилдт, М. Вильямс. SWING: руководство для начинающих (1-st Edition), 2007. – 704 с.
13. Mark Michaelis. Essential C# 6.0, 2015. – 1008 с.
14. А. Фримен. ASP.NET Core MVC с примерами на C# для профессионалов, 2017. – 992 с.
15. М. Прайс. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов, 2016. – 640 с.

16. S. Stoyan. Up & Running: Building Web Applications — Published by O’Reilly Media in Azure, 2016. – 222 с.
17. С. Морето. Bootstrap в примерах, 2016. – 316 с.
18. Р. Динеш. Все паттерны проектирования, 2019. – 320 с.

ДОДАТОК 1

Організація реєстру інформаційних ресурсів з використанням технології Blockchain

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ52131_19Б

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ ТМ52131_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ ТМ52131_19Б 12-1	/Models/*.cs	Моделі серверної частини
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ ТМ52131_19Б 12-2	/View /*.cshtml	Модуль клієнтської частини
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ ТМ52131_19Б 12-3	/Controllers/*.cs	Модуль взаємодії клієнтської та серверної частин
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ ТМ52131_19Б 13-1	Опис.docx	Опис програмного модуля

ДОДАТОК 2

Організація реєстру інформаційних ресурсів з використанням технології Blockchain

Лістинг програмного модуля

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ52131_19Б 12-1

Аркушів 10

Київ – 2019

```
/// <summary>
/// Модель блока мережі.
/// </summary>

[DataContract]
public class Block
    {
        [DataMember]
        public int Version { get; set; }

        [DataMember]
        public DateTime CreatedOn { get; set; }

        [DataMember]
        public string Hash { get; set; }

        [DataMember]
        public string PreviousHash { get; set; }

        [DataMember]
        public string Data { get; set; }

        [DataMember]
        public string User { get; set; }

        public User UserObject
        {
            get
            {
                var user = Api.Deserialize<User>(User);
                return user;
            }
        }

        public Data DataObject
        {
            get
            {
                var data = Api.Deserialize<Data>(Data);
                return data;
            }
        }

        public bool GenesisBlock
        {
            get
            {
                if(Hash == "71e12e272ea682f5e1d60fc5cf5e8ec776df33e38ac8f3f6e6b9ab5eeedde245
")
                {
                    return true;
                }

                return false;
            }
        }

        public override string ToString()
```

```
        {
            return Hash;
        }
    }

/// <summary>
/// Модель користувача.
/// </summary>

[DataContract]
public class User
    {
        [DataMember]
        public string Login { get; set; }

        [DataMember]
        public string Password { get; set; }

        [DataMember]
        public string Hash { get; set; }

        [DataMember]
        public Enums.UserRole Role { get; set; }

        public override string ToString()
        {
            return Login;
        }
    }
```



```
// Представлення сторінки авторизації

@Html.Raw(TempData["msg"])
@model Domain.Authorization

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/3.3.7/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css" />

<div class="container">
    <div class="row">

        <div class="col-md-offset-3 col-md-6" style="margin-top:20px;">
            <form class="form-horizontal" asp-controller="Authorization" asp-
action="SignIn" method="post">
                <img src="~/images/Blockchain-Funds.png" height="150" />
                <span class="heading">BLOCKCHAIN STORAGE</span>
                <div class="form-group">
                    <input asp-for="@Model.Username" type="text" class="form-
control" id="inputEmail" title="Enter username" placeholder="Username">
                    <i class="fa fa-user"></i>
                </div>
                <div class="form-group help">
                    <input asp-for="@Model.Password" type="password" class="form-
control" id="inputPassword" title="Enter password" placeholder="Password">
                    <i class="fa fa-lock"></i>
```

```
                </div>
                <div class="form-group">
                    <a class="text" asp-area="" asp-controller="Authorization" asp-
action="SignUp" title="Sign up">Sign Up</a>
                    <button type="submit" class="btn btn-
default" title="Sign in">SIGN IN</button>
                </div>
            </form>
        </div>

    </div>
</div>

// Представлення сторінки реєстрації

@Html.Raw(TempData["msg"])
@model Domain.Authorization

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/3.3.7/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css" />

<div class="container">
    <div class="row">

        <div class="col-md-offset-3 col-md-6" style="margin-top:20px;">
            <form class="form-horizontal" asp-controller="Authorization" asp-
action="SignUp" method="post">
                <img src="~/images/Blockchain-Funds.png" height="150" />
                <span class="heading">BLOCKCHAIN STORAGE</span>
                <div class="form-group">
                    <input asp-for="@Model.Username" type="text" class="form-
control" id="inputEmail" title="Enter username" placeholder="Username">
                    <i class="fa fa-user"></i>
                </div>
                <div class="form-group help">
                    <input asp-for="@Model.Password" type="password" class="form-
control" id="inputPassword" title="Enter password" placeholder="Password">
                    <i class="fa fa-lock"></i>
                </div>
                <div class="form-group help">
                    <input asp-for="@Model.ConfirmPassword" type="password" class="form-
control" id="inputPassword" title="Enter password" placeholder="Cofirm password">
                    <i class="fa fa-lock"></i>
                </div>
                <div class="form-group">
                    <a class="text" asp-area="" asp-controller="Authorization" asp-
action="SignIn" title="Sign in">Sign In</a>
                    <button type="submit" class="btn btn-
default" title="Sign up">SIGN UP</button>
                </div>
            </form>
        </div>

    </div>
</div>

// Представлення сторінки перегляду звантажених даних в мережу

@Html.Raw(TempData["msg"])
```

```
@model List<Domain.File>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand font-weight-bold" href="#">
        <img src="~/images/Blockchain.png" width="300" height="50" class="d-inline-block align-top" alt="">
        STORAGE
    </a>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
            <a class="nav-item nav-link active" href="#">Downloaded Data</a>
            <a class="nav-item nav-link" asp-area="" asp-controller="Data" asp-action="SetData" title="Sign up">Set Data</a>
            <a class="nav-item nav-link" asp-area="" asp-controller="Data" asp-action="GetData" title="Sign up">Get Data</a>
        </div>
    </div>
    <form class="form-inline mb-0">
        <div>@Blockchain.Helper.CurrentAcc.Name</div>
        <a class="btn btn-danger ml-4 btn-sm" asp-area="" asp-controller="Authorization" asp-action="SignIn" title="Exit">Exit</a>
    </form>
</nav>
<div class="container body-content">
    <div class="mx-5">
        <h3 class="my-4 mx-3">Downloaded data</h3>
        <ul class="list-group">
            @foreach (var item in @Model)
            {
                <li class="list-group-item d-flex justify-content-between align-items-center">
                    @item.Name
                    <a asp-area="" asp-controller="Data" asp-action="GetData"><span class="badge badge-secondary badge-pill">Get</span></a>
                </li>
            }
        </ul>
        <footer class="mt-5">
            <hr />
            <p>&copy; 2019 - NTUU KPI</p>
        </footer>
    </div>
</div>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></script>

// Представлення сторінки отримання даних з мережі
```

```
@Html.Raw(TempData["msg"])
@model Domain.File

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand font-weight-bold" href="#">
        <img src="~/images/Blockchain-Funds.png" width="30" height="30" class="d-inline-block align-top" alt="">
        BLOCKCHAIN STORAGE
    </a>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
            <a class="nav-item nav-link" asp-area="" asp-controller="Data" asp-action="DownloadedData" title="Sign up">Downloaded Data</a>
            <a class="nav-item nav-link" asp-area="" asp-controller="Data" asp-action="SetData" title="Sign up">Set Data</a>
            <a class="nav-item nav-link active" href="#">Get Data</a>
        </div>
    </div>
    <form class="form-inline mb-0">
        <div>@Blockchain.Helper.CurrentAcc.Name</div>
        <a class="btn btn-danger ml-4 btn-sm" asp-area="" asp-controller="Authorization" asp-action="SignIn" title="Exit">Exit</a>
    </form>
</nav>
<div class="container body-content">
    <div class="mx-5">
        <h3 class="m-4">Get data</h3>
        <form asp-controller="Data" asp-action="GetData" method="post" enctype="multipart/form-data">
            <div class="form-group help">
                <p class="m-3">Enter the path to save</p>
                <input asp-for="@Model.Name" type="text" class="form-control mx-3" id="inputPassword" title="Enter path" placeholder="Path">
            </div>
            <div class="form-group help">
                <p class="m-3">Enter key file</p>
                <input asp-for="@Model.Key" type="password" class="form-control mx-3" id="inputPassword" title="Enter key" placeholder="Key">
            </div>
            <div class="form-group mt-5 ml-3">
                <button type="submit" class="btn btn-secondary" value="upload">Set data</button>
            </div>
        </form>
        <footer class="mt-5">
            <hr />
            <p>&copy; 2019 - NTUU KPI</p>
        </footer>
    </div>
</div>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>
```

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integri
ty="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></
script>

// Представлення сторінки завантаження даних до мережі

@Html.Raw(TempData["msg"])
@model Domain.File

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstra
p.min.css" integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand font-weight-bold" href="#">
        <img src="~/images/Blockchain-Funds.png" width="30" height="30" class="d-inline-
block align-top" alt="">
        BLOCKCHAIN STORAGE
    </a>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
            <a class="nav-item nav-link" asp-area="" asp-controller="Data" asp-
action="DownloadedData" title="Sign up">Downloaded Data</a>
            <a class="nav-item nav-link active" href="#">Set Data<span class="sr-
only">(current)</span></a>
            <a class="nav-item nav-link" asp-area="" asp-controller="Data" asp-
action="GetData" title="Sign up">Get Data</a>
        </div>
    </div>
    <form class="form-inline mb-0">
        <div>@Blockchain.Helper.CurrentAcc.Name</div>
        <a class="btn btn-danger ml-4 btn-sm" asp-area="" asp-
controller="Authorization" asp-action="SignIn" title="Exit">Exit</a>
    </form>
</nav>
<div class="container body-content">
    <div class="mx-5">
        <h3 class="m-4">Set data</h3>
        <p class="m-3">Enter file for added</p>
        <form asp-controller="Data" asp-
action="SetData" method="post" enctype="multipart/form-data">
            <div class="form-group m-3">
                <label for="file" class="font-weight-bold">File:</label>
                <input type="file" name="file" class="form-control-file">
            </div>
            <p class="m-3">Enter key for file</p>
            <div class="form-group help">
                <input asp-for="@Model.Key" type="password" class="form-control mx-
3" id="inputPassword" title="Enter key" placeholder="Key">
            </div>
            <div class="form-group mt-5 ml-3">
                <button type="submit" class="btn btn-
secondary" value="upload">Set data</button>
            </div>
        </form>
        <footer class="mt-5">
            <hr />
            <p>&copy; 2019 - NTUU KPI</p>
        </footer>
    </div>
```

```
</div>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></
script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" inte
grity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></
script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integri
ty="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></
script>
```



```
// Контролер
public class ApiController : Controller
    {

        private readonly IHostingEnvironment _hostingEnvironment;

        public ApiController(IHostingEnvironment hostingEnvironment)
        {
            _hostingEnvironment = hostingEnvironment;
        }

        [HttpGet]
        public IActionResult SignIn()
        {
            return View();
        }

        [HttpGet]
        public IActionResult SignUp()
        {
            return View();
        }

        [HttpGet]
        public IActionResult SetData()
        {
            return View();
        }

        [HttpGet]
        public IActionResult GetData()
        {
            return View();
        }

        [HttpGet]
        public IActionResult DownloadedData()
        {
            return View();
        }

        [HttpPost]
        public IActionResult SignIn(Authorization authorizationData)
        {
            if (ChekcAcc(Deserialize<string>(authorizationData.ConnectionGroupId)))
```

```
            {
                return RedirectToAction("DownloadedData", "Data");
            }
            TempData["msg"] = "<script>alert('Not correct data!');</script>";
            return View();
        }

        [HttpPost]
        public IActionResult SignUp(Authorization authorizationData)
        {
            if (CreateAcc(Deserialize<string>(authorizationData.ConnectionGroupId)))
            {
                TempData["msg"] = "<script>alert('Passwords do not match!');</script>";
                return View();
            }
            if (ChekcAcc(Deserialize<string>(authorizationData.Message)))
            {
                TempData["msg"] = "<script>alert('An account already exists!');</script>";
                return View();
            }
            return RedirectToAction("DownloadedData", "Data");
        }

        public bool ChekcAcc(string data)
        {
            var json = SendRequest("chekcacc", data);
            return !string.IsNullOrEmpty(json);
        }

        public bool CreateAcc(string data)
        {
            var json = SendRequest("createacc", data);
            return !string.IsNullOrEmpty(json);
        }

        public bool AddData(string data)
        {
            var json = SendRequest("adddata", data);
            return !string.IsNullOrEmpty(json);
        }

        private string SendRequest(string method, string data)
        {
            using (var client = new HttpClient())
            {
                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue(
"application/json"));
                client.Timeout = TimeSpan.FromSeconds(20);

                // http://localhost:28451/BlockchainService.svc/api/adddata/
                string repUri = $"BlockchainService.svc/api/{method}/{data}";
                var response = client.GetAsync(repUri).Result;
                if (response.IsSuccessStatusCode)
                {
                    var result = response.Content.ReadAsStringAsync().Result;
                    return result;
                }
            }

            return null;
```

```
    }

    public static T Deserialize<T>(string json)
    {
        using (var ms = new MemoryStream(Encoding.UTF8.GetBytes(json)))
        {
            var deserializer = new DataContractJsonSerializer(typeof(T));
            var requestResult = (T)deserializer.ReadObject(ms);

            return requestResult;
        }
    }

    [HttpPost]
    public IActionResult SetData(IFormFile file)
    {
        if (file != null)
        {
            if (file.ContentType == null)
            {
                TempData["msg"] = "<script>alert('Enter key!');</script>";
                return View();
            }
            var json = SendRequest("adddata", Deserialize<string>(file.ContentType));
            TempData["msg"] = "<script>alert('File added!');</script>";
            return View();
        }
        else
        {
            TempData["msg"] = "<script>alert('Enter file!');</script>";
            return View();
        }

    }

    [HttpPost]
    public IActionResult GetData(IFormFile file)
    {
        var json = SendRequest("getdata", Deserialize<string>(file.ContentType));
        if (json.IsNormalized())
        {


            TempData["msg"] = "<script>alert('File saved!');</script>";
            return RedirectToAction("DownloadedData");
        }
        TempData["msg"] = "<script>alert('Not Found!');</script>";
        return View();
    }
}
```

ДОДАТОК 3

Організація реєстру інформаційних ресурсів з використанням технології Blockchain

Опис програмного модуля

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ52131_19Б 13-1

Аркушів 10

Київ – 2019

АНОТАЦІЯ

Даний клас надає можливість створити зв’язок між серверною частиною програмного продукту та його клієнтської частини.

Розроблений модуль дозволяє отримати дані від користувача, серилізувати та записати в мережу, і десериалізувати для збереження на власний носій.

Засіб програмної реалізації – мова програмування C#.

Середовище розробки – Visual Studio 2017.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ

Розроблений клас призначений для взаємодії клієнтської частини та бізнес логіки програмного продукту за допомогою REST-архітектури.

Клас містить GET-методи для отримання даних, POST-методи для надсилання даних та додаткові методи для валідації даних, їх серилізації та десерилізації.

Засіб програмної реалізації – мова програмування C#.

Середовище розробки – Visual Studio 2017.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Функціонування модуля обумовлене функціями, які описані в таблиці 2.1.

Таблиця 2.1. Вимоги до програмного забезпечення

Назва	Опис
SignIn ()	Завантаження сторінки авторизації
SignUp ()	Завантаження сторінки реєстрації
SetData ()	Завантаження сторінки для завантаження даних в мережу
GetData ()	Завантаження сторінки для отримання даних з мережі
DownloadedData ()	Завантаження сторінки для перегляду завантажених даних
SignIn (Authorization authorizationData)	Отримання даних від користувача для авторизації
SignUp (Authorization authorizationData)	Отримання даних від користувача для реєстрації
SetData (IFormFile file)	Отримання файлу від користувача для завантаження в мережу
GetData (IFormFile file)	Завантаження файлу на носій
ChekcAcc (string data)	Валідація даних для авторизації
CreateAcc (string data)	Валідація даних для створення власного кабінету
AddData (string data)	Валідація завантаженого файлу
SendRequest (string method, string data)	Передача даних в мережу по переданому методу

3. ВХІДНІ ДАНІ

Вхідними даними для роботи системи на першому етапі, являються значення логіну та паролю користувача для авторизації чи реєстрації в системі.

Після входу в персональний кабінет, користувач може завантажити файли будь-якого формату до мережі, так як всі вони серилізуються до єдиного формату.

4. ВИХІДНІ ДАНІ

Вихідними даними результату роботи програмного продукту, являються всі попередньо завантажені файли користувачем до мережі.